
**UNIVERSIDADE FEDERAL DO CEARÁ**

**DEPARTAMENTO DE ENGENHARIA**

**CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO**

**RICARDO VICTOR MATOS PEREIRA**

# UMA COMPARAÇÃO DAS ABORDAGENS PARA TRATAMENTO DE CONCEPT DRIFTS EM DADOS PROCESSADOS COM MACHINE LEARNING

**SOBRAL-CE**

**2023**

RICARDO VICTOR MATOS PEREIRA

# UMA COMPARAÇÃO DAS ABORDAGENS PARA TRATAMENTO DE CONCEPT DRIFTS EM DADOS PROCESSADOS COM MACHINE LEARNING

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do  da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Wendley S. Silva.

SOBRAL-CE

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

---

P495c Pereira, Ricardo Victor Matos.
UMA COMPARAÇÃO DAS ABORDAGENS PARA TRATAMENTO DE CONCEPT DRIFTS EM DADOS PROCESSADOS COM MACHINE LEARNING / Ricardo Victor Matos Pereira. – 2023.
30 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral, Curso de Engenharia da Computação, Sobral, 2023.
Orientação: Prof. Dr. Wendley Souza da Silva.

1. aprendizado de máquina. 2. desvio de conceito. 3. detecção de desvio de conceito. I. Título.
CDD 621.39

---

## RESUMO

Este trabalho apresenta uma comparação das abordagens para o tratamento de desvios de conceito, do inglês *concept drift*, em dados processados com técnicas de machine learning. Os desvios de conceito são mudanças na distribuição dos dados ao longo do tempo, o que pode afetar o desempenho dos modelos de aprendizado de máquina. O objetivo deste estudo é analisar as estratégias para lidar com desvios de conceito e avaliar sua eficácia na manutenção do desempenho dos modelos ao longo do tempo. Foram consideradas abordagens baseadas como retreinamento, retreinamento usando métodos *ensemble* e algoritmos adaptativos. A metodologia envolveu a implementação e experimentação das abordagens em um conjunto de dados real, além de análises comparativas dos resultados obtidos. Os resultados indicam que cada abordagem possui vantagens e desvantagens dependendo do cenário e características dos dados, e não há uma solução única que se aplique a todos os casos. Essa comparação fornece informações valiosas para a seleção adequada de técnicas de tratamento de desvios de conceito em problemas de aprendizado de máquina.



**Palavras-chave**: aprendizado de máquina; desvio de conceito; detecção de desvio de conceito.

## ABSTRACT

This work presents a comparison of approaches for handling concept drift in machine learning models applied to processed data. Concept drift refers to changes in the data distribution over time, which can impact the performance of machine learning models. The objective of this study is to analyze strategies for dealing with concept drift and evaluate their effectiveness in maintaining model performance over time. The considered approaches include retraining, ensemble-based retraining, and adaptive algorithms. The methodology involved implementing and experimenting with these approaches on a real dataset, as well as conducting comparative analyses of the results. The findings indicate that each approach has its own advantages and disadvantages depending on the scenario and data characteristics, and there is no one-size-fits-all solution. This comparison provides valuable insights for the appropriate selection of concept drift handling techniques in machine learning problems.



**Keywords**: machine learning; concept drift; concept drift detection.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| ADWIN | *Adaptative Windowing* |
| CM | *Competence Model-based drift detection* |
| DDM | *Drift Detection Method* |
| IA | Inteligência Artificial |
| IoT | Internet of Things |
| ML | *Machine Learning* |
| SCD | *Statistical Change Detection for multi-dimensional data* |
| SGD | *Stochastic Gradient Descent* |
| SVM | *Support Vector Machine* |

**SUMÁRIO**

# 1 INTRODUÇÃO

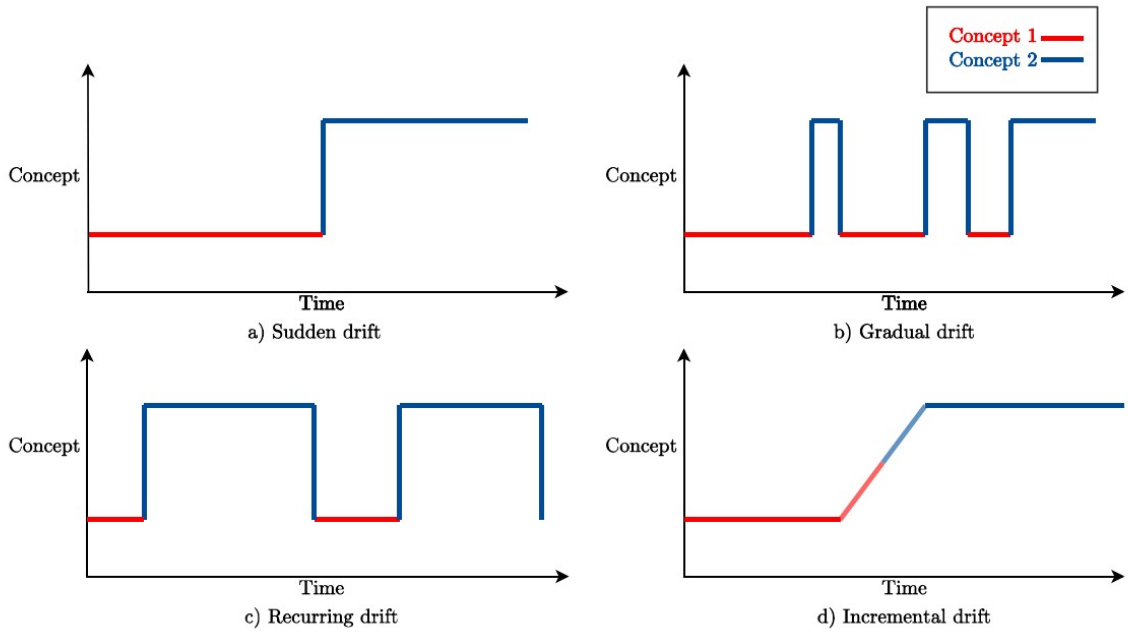
Nos últimos anos a quantidade de dados gerados mundialmente chegam a valores exorbitantes. As informações que podem ser obtidas a partir desses dados através da utilização de ML são de alta relevância para grandes empresas ou governos para realizar previsões e tomar decisões. Estes dados podem ser obtidos de diversas formas, por exemplo, controle de tráfego aéreo, rastreamento de dispositivos móveis, medição de sensores, frequência de internações médicas em um hospital, entre outros.

Mesmo utilizando técnicas de ML para aprender sobre um conjunto de dados, a evolução constante dos padrões e modelos utilizados nesses aprendizados faz com que erros nas predições aconteçam, levando o estudo a um resultado não desejado. Esses comportamentos indesejados são categorizados como, do inglês, *concept drifts*. Quando um *concept drift* ocorre, o padrão induzido de dados antigos pode não ser relevante para os novos dados, levando a previsões e resultados de decisão ruins. O fenômeno de *concept drifts* tem sido reconhecido como a causa raiz da diminuição de efetividade de sistemas de informação, alerta e suporte. Em ambientes de constantes mudanças e grande quantidade de dados, como fornecer predições baseadas em dados mais confiáveis e decisões mais fáceis, o estudo deste fenômeno tornou-se uma questão crucial (LU *et al.*, 2019).

Segundo LU *et al.*, (2019), estudos relacionados à *concept drift* estão sendo realizados para desenvolver metodologias, técnicas para detecção de *drifts*, adaptações e melhor compreensão do problema. Através desses estudos será possível melhorar o aprendizado de dados que sofrem mudanças constantes, aprimorando o resultados dos treinamentos realizados pelos modelos de ML.

O avanço rápido e contínuo na geração e coleta de dados tem levado ao surgimento de desafios significativos na aplicação de técnicas de machine learning. Um desses desafios é o conceito de *drift*, que ocorre quando a distribuição dos dados muda ao longo do tempo, afetando negativamente o desempenho dos modelos de machine learning treinados anteriormente. A Figura 1 demonstra esses tipos de mudanças que ocasionam os desvios, são eles, desvio repentino, gradual, recorrente e incremental. O tratamento eficaz de concept drifts é crucial para garantir a confiabilidade e precisão dos modelos em ambientes dinâmicos. Portanto, é essencial realizar uma comparação das abordagens disponíveis para lidar com concept drifts em dados processados com machine learning, a fim de identificar as técnicas mais eficientes e adequadas para enfrentar esse problema.

Figura 1 – Tipos de *concept drifts*




Fonte: (BAYRAM *et al.*, 2022).

## 1.1 Objetivos

O objetivo dessa pesquisa é analisar os efeitos de uma detecção de *concept drift* após o retreino da base de dados utilizando uma técnica de adaptação do modelo de ML. De forma mais específica, realizar uma revisão abrangente da literatura relacionada a *concept drifts*, explorando as principais abordagens propostas para tratar esse problema, identificando e selecionando as abordagens mais relevantes e amplamente utilizadas para o tratamento desses *concept drifts*, considerando critérios como eficácia e eficiência. Com esse objetivo alcançado poderemos verificar se existe uma melhoria na predição após realizar o tratamento de *concept drift*.

# 2 FUNDAMENTAÇÃO TEÓRICA

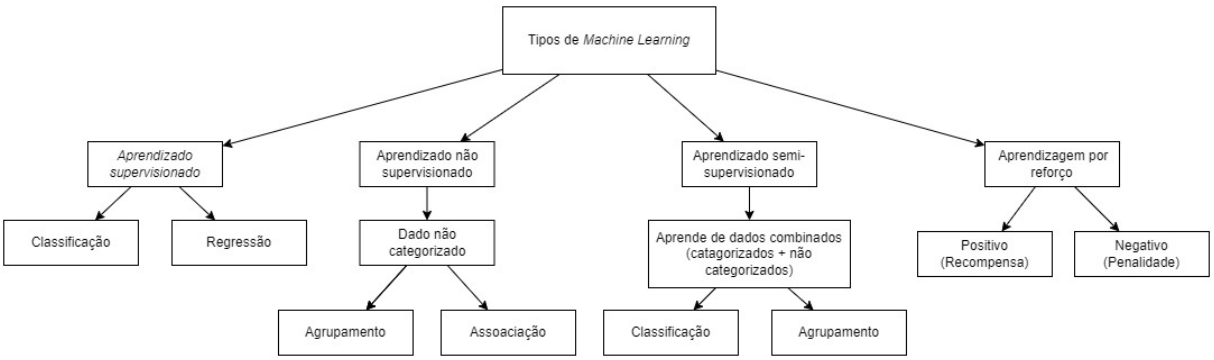
Esse capítulo abordará alguns conceitos importantes que irão facilitar o entendimento do conteúdo que será utilizado para realizar o Trabalho de Conclusão de Curso. Na Seção 2.1 será introduzido o conceito de ML. Nas seções 2.1.1, 2.1.2, 2.1.3 e 2.1.4 serão citadas as categorias de aprendizado, as seções 2.1.5 e 2.1.6 descrevem os modelos que serão utilizados nos treinamentos. Na Seção 2.2 será conceituada a ideia de *concept drift*.

## 2.1 *Machine Learning*

De acordo com SAMUEL, (1959), ML é definido como a área de estudo que permite aos computadores a capacidade de aprender sem serem explicitamente programados. ML é usado para ensinar máquinas a como lidar com dados com mais eficiência. Às vezes, depois de visualizar os dados, não podemos interpretar as informações extraídas dos dados. Nesse caso, aplicamos ML (MAHESH, 2020). Utilizando algoritmos de ML, a partir de uma amostra de dados é possível treinar essa amostra para que seja construído um modelo que irá possibilitar a análise de predições e decisões sobre o conjunto.

Conforme citado por (MOHAMMED *et al.*, 2016 apud SARKER, 2021), algoritmos de ML são divididos principalmente em quatro categorias: aprendizado supervisionado, aprendizado não supervisionado, aprendizado semi-supervisionado e aprendizagem por reforço. Na Figura 2 são apresentados alguns exemplos dos vários tipos e tecnicas englobadas por essas categorias.

Figura 2 – Tipos de técnicas de ML




Fonte: Adaptado de (SARKER, 2021).

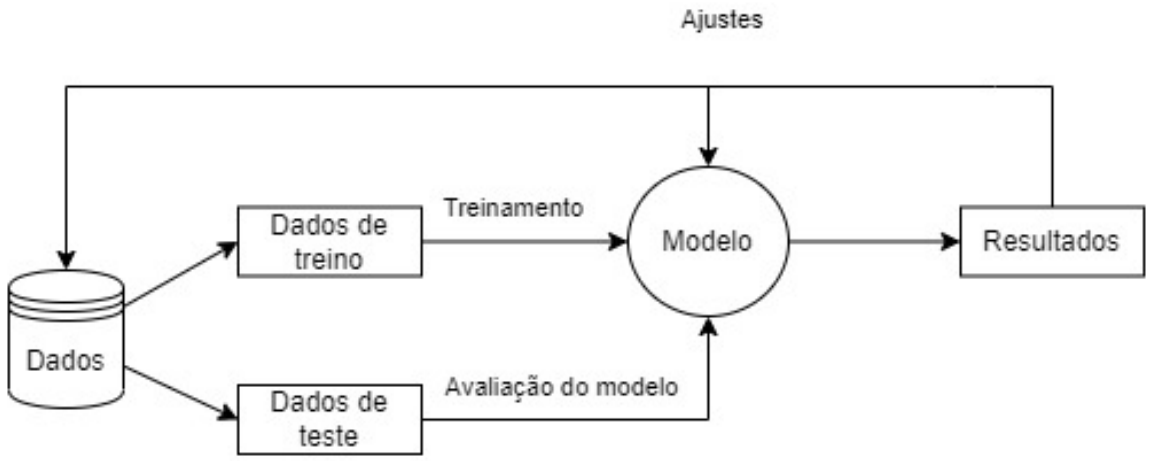
### *2.1.1 Aprendizado supervisionado*

Aprendizado supervisionado é um modelo de aprendizado que consiste em aprender uma função que mapeia uma entrada e uma saída tendo como base um exemplo de entrada-saída. O modelo infere uma função a partir dos dados de treino que consistem um conjunto de exemplos representativos do problema (MAHESH, 2020). Casos comuns de aprendizados supervisionados são:

- **Classificação**: As entradas do treinamento são divididas em duas ou mais classes, o algoritmo deverá produzir um modelo que classifique entradas ainda não treinadas em uma ou mais de uma (classificação multi-categorizada) dessas classes e preveja se algo pertence ou não a uma classe específica. Isso geralmente é feito de uma forma supervisionada. Modelos de classificação podem ser categorizados em dois grupos: classificação binária e classificação multiclasse. O filtro de *spam* é um exemplo de classificação binária, onde as entradas são os *email's* e as classes são "*spam*"ou "não *spam*"(GEEKS, 2022).
- **Regressão**: É também um problema de aprendizado supervisionado, que prevê um valor numérico e as saídas são contínuas em vez de discretas. Pode-se citar, por exemplo, a previsão de preços das ações utilizando dados históricos. (GEEKS, 2022).

Uma exemplificação de fluxo dessa categoria é demonstrada na Figura 3.

Figura 3 – Fluxo aprendizado supervisionado.




Fonte: Adaptado de (MAHESH, 2020).

### *2.1.2 Aprendizado não supervisionado*

Essa categoria é chamada de aprendizagem não supervisionada porque, ao contrário de aprendizagem supervisionada as saídas não são apresentadas ou são inexistentes durante o treinamento e não há dados de treino. Algoritmos são deixados por conta própria para descobrir e apresentar a estrutura presente nos dados. Quando novos dados são introduzidos, ele usa o recursos previamente aprendidos para reconhecer a classe dos dados. É usado principalmente para agrupamentos e redução de recursos (MAHESH, 2020).

### *2.1.3 Aprendizado semi-supervisionado*

Considerado uma combinação do aprendizado supervisionado e o não supervisionado. Utilizado geralmente onde o conjunto de dados é bastante grande e alguns se encontram categorizados e outros não.

O objetivo principal de um modelo de aprendizado semi-supervisionado é fornecer um ótimo resultado para a previsão do que o produzido utilizando apenas os dados categorizados do modelo. Algumas áreas de aplicação em que o aprendizado semi-supervisionado é usado incluem tradução, detecção de fraude, categorização de dados e classificação de texto (SARKER, 2021).

### *2.1.4 Aprendizado por reforço*

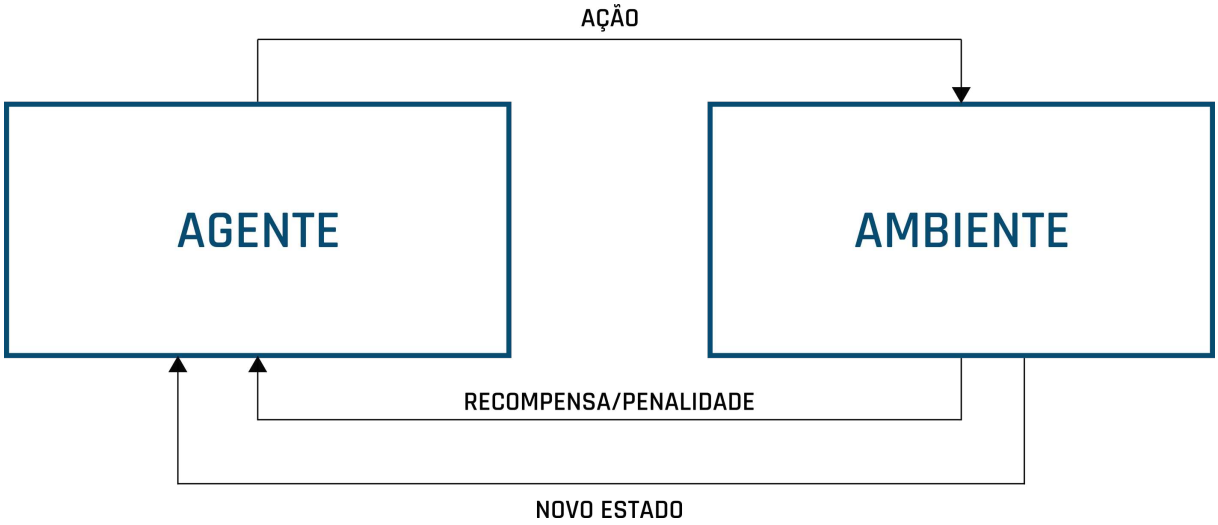
Um algoritmo que utiliza aprendizado por reforço como modelo busca atingir um certo objetivo através da metodologia de recompensa ou penalidade, onde o sistema interage com um ambiente dinâmico.

Considerada como uma ferramenta poderosa para treinar modelos de Inteligência Artificial (IA) que ajudam a aumentar a automatização ou otimização da eficiência de sistemas róboticos, tarefas de direção autônoma, fabricação e logística da cadeia de suprimentos, mas não ideal para resolver problemas básicos (SARKER, 2021). A Figura 4 contém uma exemplificação de como funciona o fluxo do modelo.

### *2.1.5 Modelo de Aprendizado Perceptron*

O modelo *perceptron* é uma representação simplificada de um neurônio biológico. Ele pega um conjunto de entradas, aplica pesos a elas e produz uma saída com base em uma

Figura 4 – Representação do fluxo de aprendizado por reforço




Fonte: (DATAAT, 2022).

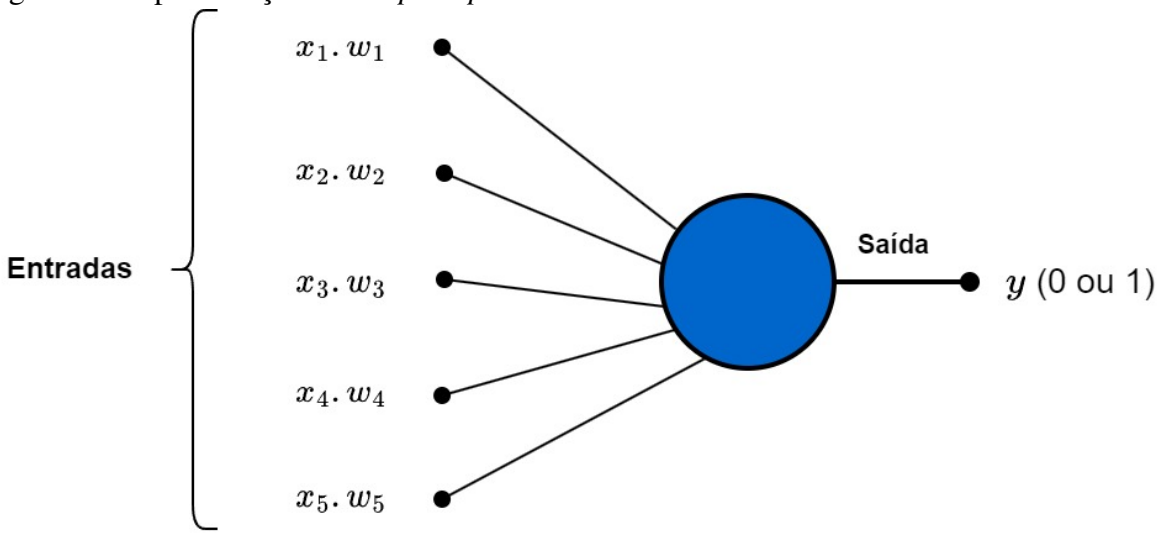
função de ativação. O algoritmo de aprendizado baseado neste modelo é um algoritmo de aprendizado supervisionado. Este ajusta os pesos do *perceptron* com base no erro entre a saída prevista e a saída alvo, visando minimizar o erro (ROSENBLATT, 1958). O algoritmo computa a soma ponderada de todas as coordenadas do vetor de padrões gerando um vetor de sáida +1 ou 0 que ficam abaixo ou acima do limiar se maior ou menor que esse limiar. Cada entrada recebida pelo *perceptron* possui um peso que depende do seu grau de contribuição na obteção da saída final. O bias *(b)* representado na equação 2.1, nos permite mover a linha de decisão com o objetivo de separar melhor a entrada em duas classes (BEGUM *et al.*, 2019).

$$f(x) = \begin{cases} 1 & \text{se } w \cdot +b > 0 \\ 0 & \text{se } w \cdot x + b \leq 0 \end{cases} . \tag{2.1}$$

, onde $w =$ vetor de pesos das entradas, $x =$ vetor de entradas, $b =$ bias e $f(x) =$ função das saídas desejadas.

O processo de treinamento de um modelo *perceptron* consiste em fazer com que o modelo aprenda os valores ideais de pesos e bias. Apresentamos ao modelo os dados de entrada e as possíveis saídas, treinamos o modelo e com isso os pesos e bias são aprendidos. Com o modelo treinado, podemos apresentar novos dados de entrada e o modelo será capaz de prever a saída (DATASCIENCEACADEMY, 2022).

Figura 5 – Representação de um *perceptron*.




Fonte: Adaptado de (DATASCIENCEACADEMY, 2022).

### *2.1.6 Modelo de Aprendizado Support Vector Machines*

As *Support Vector Machine* (SVM)s são um poderoso algoritmo de aprendizado de máquina supervisionado que tem sido amplamente utilizado em várias aplicações de classificação e regressão. As SVMs foram originalmente propostas por (CORTES; VAPNIK, 1995)Vapnik et al. (1995) como um método eficaz para resolver problemas de classificação binária, mas foram posteriormente estendidas para problemas de classificação multiclasse e regressão.

O conceito fundamental por trás das SVMs é encontrar um hiperplano de separação ótimo entre as classes, maximizando a margem entre as amostras de treinamento mais próximas a ele. Essas amostras, chamadas de vetores de suporte, são os pontos de dados críticos que definem a posição do hiperplano de separação. A principal vantagem das SVMs é a capacidade de generalização, permitindo lidar com dados não linearmente separáveis por meio de truques de kernel.

O processo de treinamento de uma SVM envolve a resolução de um problema de otimização convexa, onde o objetivo é minimizar a função custo, que penaliza a violação da margem pelos exemplos de treinamento.

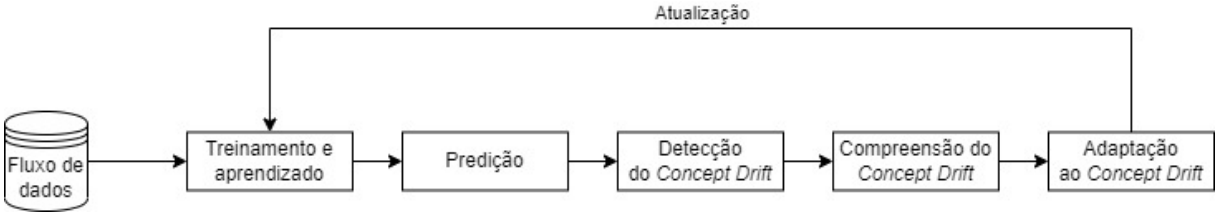
## 2.2 *Concept Drift*

Conforme WIDMER e KUBAT (1996, apud (LU *et al.*, 2019)) cita, *concept drift* é definido como um fenômeno que ocorre quando as propriedades estatísticas de uma variável

desejada se altera durante um período de tempo de várias formas, fazendo com que os dados sofram alterações levando a um resultado não esperado. Essas alterações inesperadas ocorrem devido ao tamanho muito grande de dados disponíveis que vão sendo atualizados com o decorrer do tempo, fazendo com que novas características apareçam e prejudicando o resultado final desejado. Sendo um problema bem recente, estudos na área de *concept drift* estão sendo desenvolvidos para solucionar alguns problemas, tais como detectar de forma exata um *concept drift* em uma base de dados ruidosa ou como reagir eficientemente à um *concept drift*.

O fluxo de desenvolvimento e utilização de modelos de aprendizado de máquinas pode ser dividido em duas fases: a fase de treinamento ou aprendizado, e a predição do resultado. De acordo com pesquisas mais recentes, dentro de uma ocorrência de *concept drift* são adicionados mais três componentes a esse fluxo, a detecção do desvio, o entendimento da ocorrência desse desvio (quando, como e onde) e a adaptação após a detecção desse desvio (LU *et al.*, 2019). Na Figura 6 se encontra uma representação desse novo fluxo.

Figura 6 – Representação do fluxo de aprendizado com a ocorrência de um *concept drift*.




Fonte: Adaptado de (LU *et al.*, 2019).

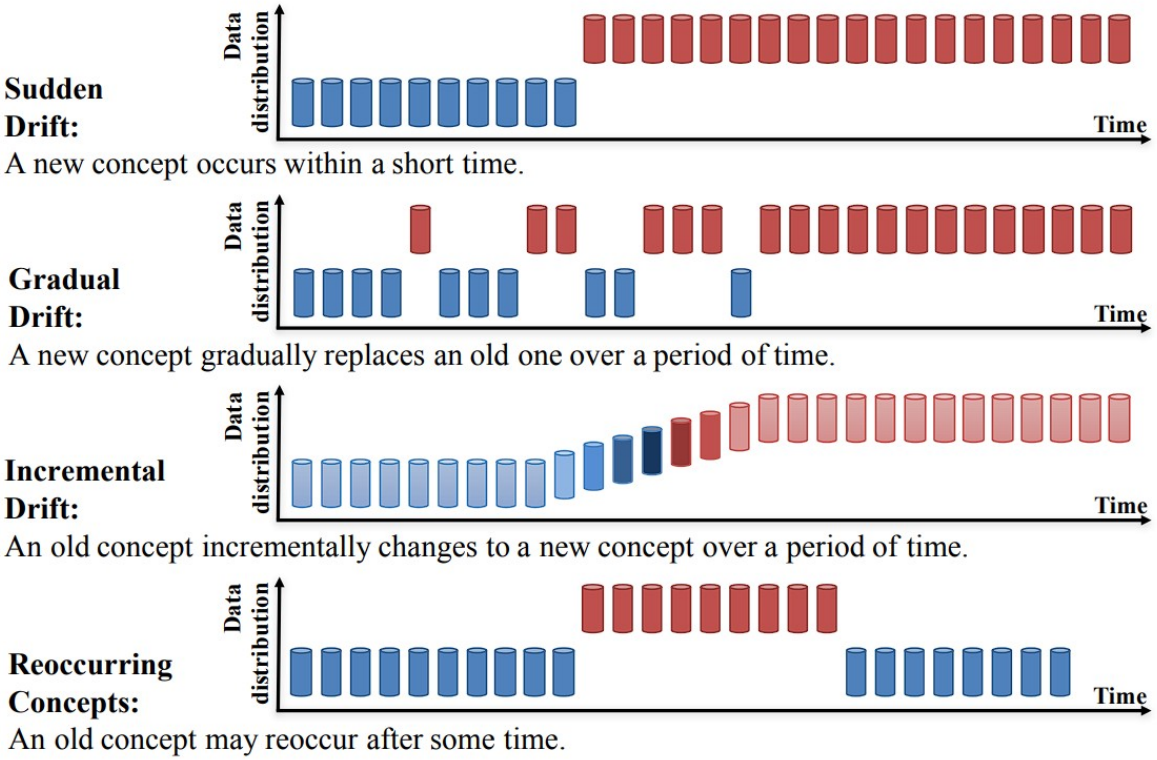
### *2.2.1 Tipos de concept drift*

Os *concept drifts* podem ser classificados em quatro tipos:

- **Desvio repentino (*Sudden drift*)**: Um novo conceito ocorre após um curto período de tempo;
- **Desvio progressivo (*Gradual drift*)**: Um novo conceito substitui um antigo durante um período de tempo;
- **Desvio incremental (*Incremental drift*)**: Um conceito antigo muda incrementalmente para um novo conceito durante um período de tempo;
- **Desvio recorrente (*Reoccurring concepts*)**: Um conceito antigo pode recorrer após um período de tempo;

Na Figura 7 temos uma representação em gráfico dos tipos existentes de *concept drift*.

Figura 7 – Representação dos tipos de *concept drift*.




Fonte: (LU *et al.*, 2019).

### *2.2.2 Detecção de concept drift*

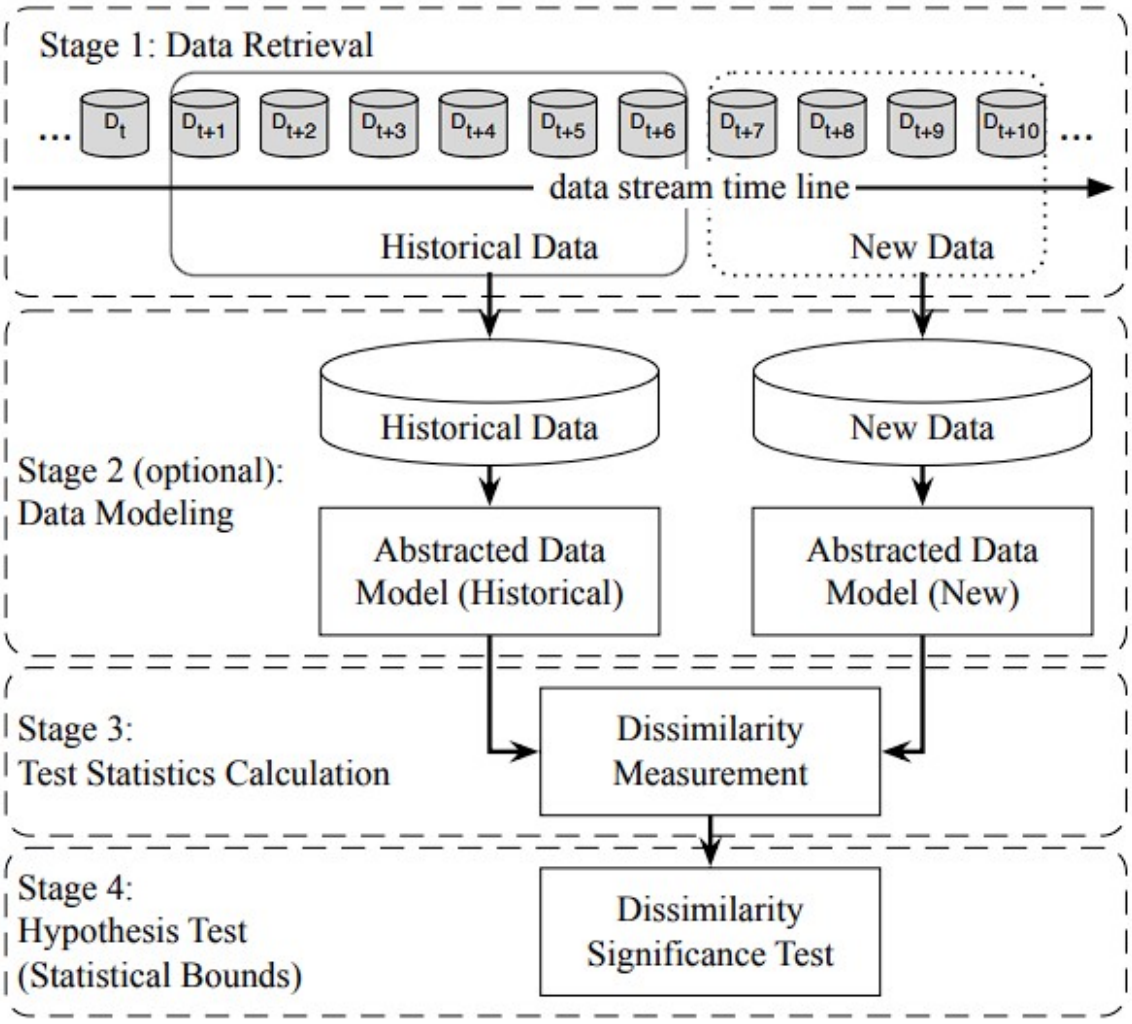
A detecção de *concept drift* tem como objetivo compilar um conjunto de técnicas e mecanismos que possuem a finalidade de caracterizar e quantificar os desvios através da identificação dos pontos ou intervalos de tempo em que esses desvios ocorrem (LU *et al.*, 2019). A Figura 8 possui uma representação de um *framework* genérico utilizado para elaboração de ferramentas que buscam identificar os desvios em uma base de dados, usualmente dividida em quatro estágios conforme descritas abaixo:

- **Estágio 1 (Recuperação dos dados)**: Tem como objetivo recuperar bloco de dados dentro de um fluxo de dados para serem utilizados nas tarefas de análises, é necessário saber identificar um conjunto de dados relevante que possuem características significativas para as observações que irão ser realizadas;
- **Estágio 2 (Modelagem dos dados)**: Nesse estágio será realizado um filtro nos recursos que possuem informações sensíveis dentro do conjunto de dados selecionado, tais recursos serão extraídos devido o impacto que eles causam caso um desvio ocorra;
- **Estágio 3 (Cálculo de estatísticas de teste)**: Tem como papel dimensionar o tamanho

do impacto que o desvio irá afetar, a partir disso irá formar estatísticas de teste que serão utilizados pelo próximo estágio. Devido ao fato de definir a gravidade do desvio no sistema, esse estágio possui um aspecto desafiador para determinar uma medida de diferenciação robusta e precisa;

– **Estágio 4 (Teste de hipótese)**: Será responsável por aplicar testes de hipótese nos resultados das diferenciações observadas no estágio anterior a fim de avaliar se as propriedades estatísticas propostas são significativas. Basicamente, sem esse estágio os testes estatísticos definidos no estágio 3 não terão serventia alguma já que não será possível determinar um intervalo de confiança onde o desvio ocorre, ou seja, se a mudança foi causada por um *concept drift* ou se foi somente uma amostra aleatória que não ocasiona alguma interferência no sistema;

Figura 8 – Framework detecção de *concept drift*.




Fonte: (LU *et al.*, 2019).

### *2.2.3 Categorias de algoritmos para detecção de concept drift*

Tendo como referência o fremework apresentado na Figura 8, os algoritmos de detecção de *concept drift* possuem três categorias e se diferenciam basicamente a partir do teste estatístico que eles aplicam para realizar a detecção.

#### *2.2.3.1 Detecção de desvio baseado em taxa de erro*

Considerada a categoria mais abrangente em quantidade de algoritmos, tem o foco em rastrear as modificações das taxas de erro que acontecem em tempo real nos classificadores do sistema. Se um aumento ou uma queda da taxa é provada estatisticamente relevante, um alarme é acionado e retornado que o desvio foi detectado (LU *et al.*, 2019). Alguns exemplos de algoritmos dessa categoria são o *Drift Detection Method* (DDM) e o *Adaptative Windowing* (ADWIN).

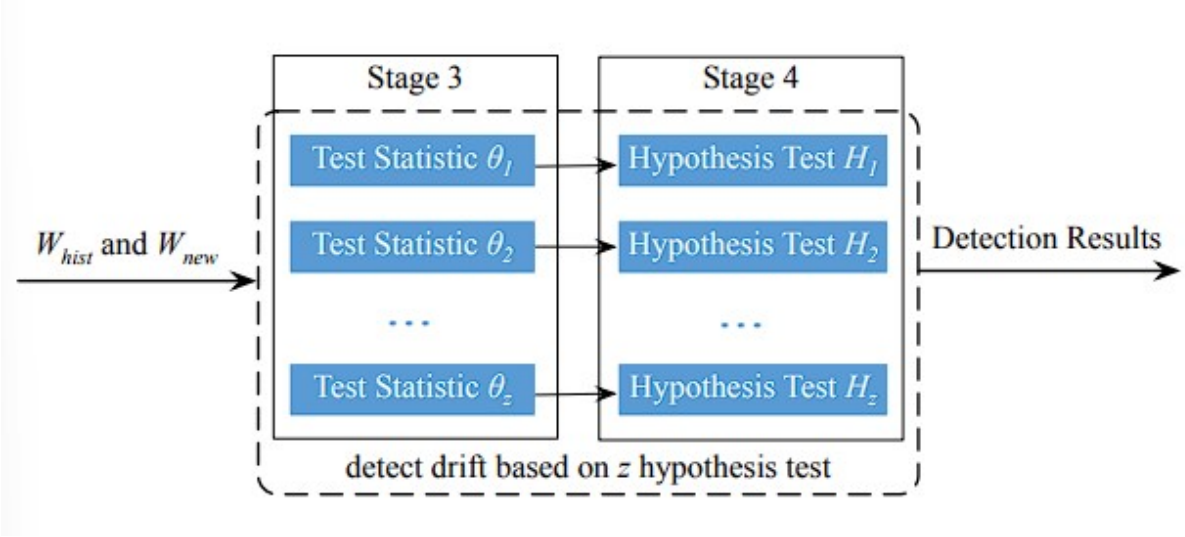
#### *2.2.3.2 Detecção de desvio baseado distribuição de dados*

Algoritmos dessa categoria utilizam uma métrica de distanciamento para dimensionar a diferenciação entre os dados antigos e os novos dados que estão sendo utilizados no sistema. Caso essa diferenciação seja provada estatisticamente,o algoritmo irá acionar um processo de atualização do modelo. Além de identificar com precisão o tempo em que o desvio ocorre, essa categoria também consegue fornecer informações sobre a localização do desvio (LU *et al.*, 2019). *Statistical Change Detection for multi-dimensional data* (SCD) e *Competence Model-based drift detection* (CM) são exemplos de algoritmos que representam essa categoria.

#### *2.2.3.3 Detecção de desvio por teste de hipótese múltipla*

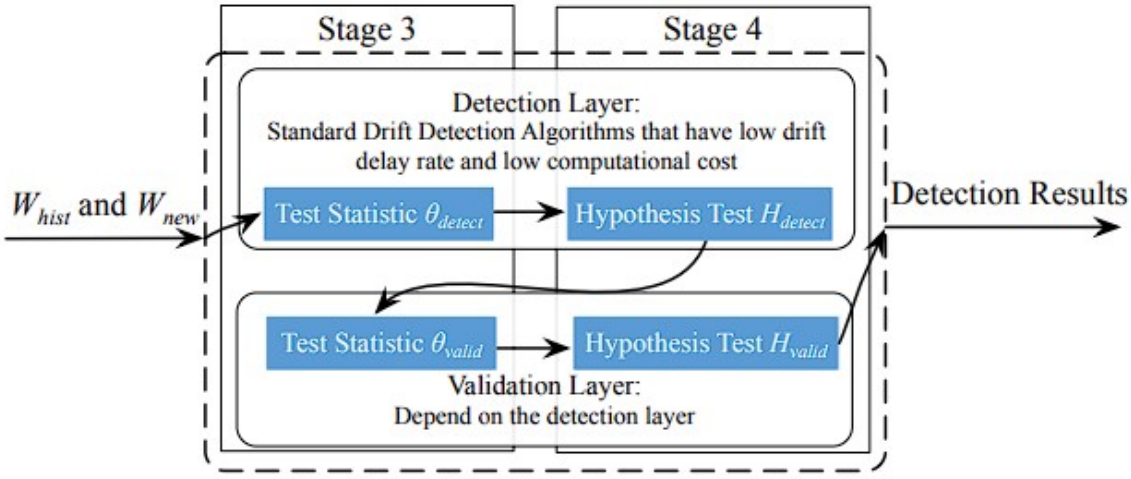
Tal categoria aplica técnicas parecidas utilizadas pelas categorias anteriores citadas, a diferença seria que nessa são utilizados múltiplos testes de hipótese para detectar os *concept drifts*. Os algoritmos são divididos em dois grupos: múltiplos testes paralelos e múltiplos testes hierárquicos. O comportamento desses grupos podem ser visualizados nas Figuras 9 e 10 (LU *et al.*, 2019).

Figura 9 – Detecção de desvio por múltiplos testes de hipótese paralelos.




Fonte: (LU *et al.*, 2019).

Figura 10 – Detecção de desvio por multiplos testes de hipótese hierárquicos.




Fonte: (LU *et al.*, 2019).

### *2.2.4 Adaptações ao drift*

As adaptações ao *concept drift* são técnicas e estratégias utilizadas para lidar com as mudanças de conceito em dados processados com machine learning. Essas adaptações visam garantir que os modelos de aprendizado sejam capazes de se ajustar às mudanças nos padrões e distribuições dos dados ao longo do tempo. Existem três grupos principais de técnicas de adpatação ao desvio, são elas:

– **Retreinamento**: envolve a reexecução do processo de treinamento do modelo utilizando dados atualizados. Isso permite que o modelo se adapte às novas informações e aprenda os

padrões mais recentes nos dados;

- **Retreinamento utilizando técnicas *ensemble***: consiste em combinar múltiplos modelos para melhorar a robustez e a capacidade de adaptação do sistema;
- **Ajuste de modelo**: baseia-se no desenvolvimento de um modelo que aprende de forma adaptativa com os dados em mudança. Tal modelo têm a capacidade de se atualizar parcialmente quando ocorre alterações de distribuição de dados (LU *et al.*, 2019);

É importante ressaltar que a seleção da estratégia mais adequada para tratar o *concept drift* depende do contexto específico e das características dos dados e do problema em questão. Diversos estudos e pesquisas têm sido realizados para investigar e comparar essas diferentes abordagens.

# 3 METODOLOGIA

A elaboração dessa pesquisa teve como finalidade realizar um comparativo entre os resultados obtidos de um algoritmo de ML após a detecção de um *concept drift*, a fim de verificar se os resultados são mais satisfatórios quando o usuário realiza ou não, alguma reação após a ocorrência do desvio.

O tipo de pesquisa foi do tipo exploratória, que segundo (VERGARA, 2013), a pesquisa exploratória é realizada em uma área que possui pouco conhecimento acumulado e sintetizado, sendo realizadas em uma área que se pretende trazer mais conhecimento e proporcionar uma visão mais explícita sobre determinado assunto. Podendo concluir que a pesquisa realizada busca se aprofundar nos conceitos referentes a *concept drift*.

## 3.1 *Dataset*

Foi realizado uma busca de um conjunto de dados na plataforma *Kaggle* (KAGGLE, 2022), onde tais dados devem possuir um fluxo temporal de captura já que o desvio será calculado comparando os dados em um determinado período de tempo.

Os dados selecionados para o experimento foram um conjunto de leituras de temperatura realizados por dispositivos de Internet of Things (IoT) instalados dentro e fora de uma sala em um intervalo considerável de tempo possuindo um total de 97.606 instâncias[1].

## 3.2 Pré-processamento dos dados

O conjunto de dados possui algumas informações que não serão necessárias para o treinamento. A Tabela 1 demonstra o conjunto de atributos que cada instância do conjunto possui. Foram removidas as informações do id, identificação da sala e data da leitura, deixando portanto somente as leituras das temperaturas como nosso vetor de entradas e o identificador onde a leitura foi realizada como nosso vetor de saída desejadas.

O vetor de saídas não está em uma estrutura ideal para a realização do treinamento, sendo assim foi realizado uma codificação binária das saídas utilizando a classe *LabelEnconder* da biblioteca *scikit-learn*, onde os valores que possuem o valor *in* serão representandos por 0 e os valores *out* serão representados por 1.

[1] Link para o dataset: https://www.kaggle.com/datasets/atulanandjha/temperature-readings-iot-devices

Tabela 1 – Representação da estrutura dos dados

| Atributo | Tipo de dado | Descrição |
|---|---|---|
| id | texto | identificadores únicos de cada leitura |
| `room_id` | texto | identificador da sala onde o dispositivo está instalado |
| `noted_date` | texto | data e tempo em que a leitura foi realizada |
| temp | inteiro | leitura da temperatura em graus celsius |
| out/in | texto | indentifica se a leitura foi realizada dentro ou fora da sala |

Fonte: elaborada pelo autor.

## 3.3 Treinamento modelo de aprendizado de máquina e predição

O processo de treinamento será responsável por criar os modelos de classificação para a futura predição de onde a leitura da temperatura foi realizada. Nesta etapa, os algoritmos de aprendizado constroem os modelos baseados no conjunto de treinamento formado pelo vetor de dados e seus rótulos correspondentes. No modelo adotado o vetor de características é representado pelas leituras das temperaturas e a classe alvo como onde essa leitura foi realizada, dentro ou fora da sala.

Das 97.606 instâncias do conjunto foi separado 70% para servir de base para este treinamento e 30% para ser utilizado como um fluxo de dados real na alimentação do algoritmo de detecção do *concept drift*. Dos 70% da base, foi separado 70% para o treinamento e 30% para predição dos modelos. A Tabela 2 informa a quantidade de dados utilizados em cada etapa.

Tabela 2 – Representação da quantidade de dados utilizados em cada etapa.

| Referência | Quantidade de instâncias |
|---|---|
| Dataset | 97.606 |
| Base de treinamento | 47.826 |
| Base detecção *concept drif* | 29.282 |
| Base para teste (predição) | 20.498 |

Fonte: elaborada pelo autor.

Para este experimento foram utilizadas duas técnicas de classificação para comparação de resultados dos treinamentos. Como os modelos devem possuir um método de treinamento disponibilizada pela biblioteca *scikit-learn* que permita o treinamento parcial dos dados, as técnicas escolhidas foram o modelo *Perceptron* e o modelo SVM. No modelo SVM foi necessário utilizar a classe *Stochastic Gradient Descent* (SGD) em conjunto para poder treinar parcialmente os dados. O SGD nada mais é que uma técnica de otimização e não corresponde a um modelo específico. No SGD, em vez de usar todo o conjunto de dados para cada iteração, apenas um único exemplo de treinamento aleatório (ou um pequeno lote) é selecionado para calcular o

gradiente e atualizar os parâmetros do modelo. Essa seleção aleatória introduz aleatoriedade no processo de otimização, daí o termo “estocástico” (GEEKS, 2023).

Ao final do treinamento é realizado a predição do modelo sem a interferência do *concept drift*, usando a base de teste informada pela Tabela 2.

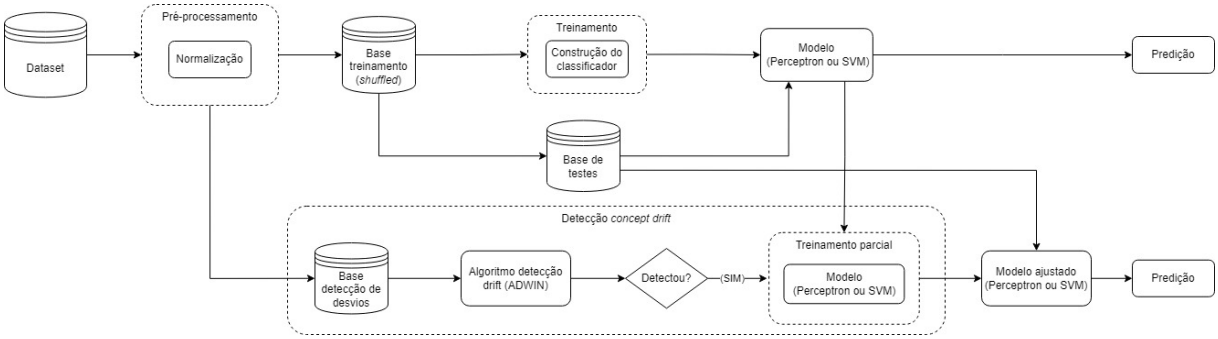
## 3.4 Detecção *Concept Drift*

A técnica de adaptação ao *concept drift* será o ajuste do modelo de treinamento ao detectar o *drift*. A detecção do *drift* ficará responsável pelo algoritmo ADWIN disponibilizado pelo pacote *scikit-multflow*. Na etapa da detecção, o algoritmo irá percorrer a base de dados referente a detecção representada na Tabela 2, toda vez que uma mudança for detectada o módulo de treinamento é acionado para realizar uma nova aprendizagem parcial utilizando os previsores do novo padrão. Ao final, após o retreino do modelo a cada detecção, é realizado uma predição do modelo usando a base de teste informada na Tabela 2.

## 3.5 Estruturação final do algoritmo

A Figura 11 apresenta o fluxograma com a estrutura final do algoritmo, tal algoritmo será executado 50 vezes, armazenando em cada passo os valores de desempenho dos modelos em cada situação, sem tratativa e com trativa da detecção dos *concept drifts*.

Figura 11 – Fluxograma algoritmo final




Fonte: elaborada pelo autor.

# 4 RESULTADOS

Após cada treino e predição realizada, foi realizado uma análise de desempenho utilizando a métrica de acurácia e o *F1-score* para comparar o sistema utilizando os algoritmos de detecção de *drift* em relação à sua não utilização. A métrica de acurácia é uma medida estatística que avalia a capacidade de um modelo ou algoritmo em classificar corretamente os dados em relação ao total de amostras avaliadas. Essa métrica é comumente utilizada em problemas de classificação, onde o objetivo é atribuir uma classe ou categoria específica a cada instância de dados. É calculada pela equação

$$acurácia = \frac{T_p + T_n}{T_p + F_p + T_n + F_n}. \tag{4.1}$$

onde $T_p$ são os verdadeiros positivos, $T_n$ são os verdadeiros negativos, $F_p$ são os falsos positivos e $F_n$ sao os falsos negativos. Já o *F1-score* é uma métrica de avaliação comumente utilizada em problemas de classificação para medir a qualidade geral de um modelo. Essa métrica considera tanto a precisão quanto a revocação do modelo. A precisão é a proporção de verdadeiros positivos em relação à soma de verdadeiros positivos e falsos positivos e a revocação é a proporção de verdadeiros positivos em relação à soma de verdadeiros positivos e falsos negativos.O *F1-score* pode ser representado pela seguinte equação

$$f1 = 2 * \frac{precisão * revocação}{precisão + revocação}. \tag{4.2}$$

sendo,

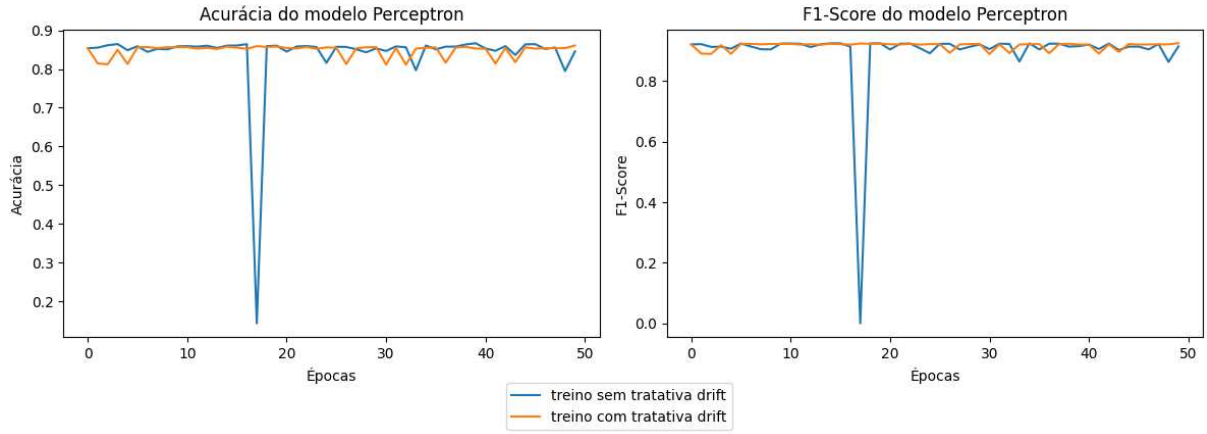
$$precisão = \frac{T_p}{(T_p + F_p)}. \tag{4.3}$$

$$revocação = \frac{T_p}{(T_p + F_n)}. \tag{4.4}$$

onde $T_p$ são os verdadeiros positivos, $F_p$ são os falsos positivos e $F_n$ os falsos negativos.

A Figura 12 apresenta os resultados do classificador utilizando o modelo *Perceptron* usando o algoritmo de detecção de *drift* em comparação ao seu não uso nas 50 repetições realizadas. O algoritmo de detecção detectou 120 desvios na base de dados de detecção. A Tabela 3 informa a média e o desvio padrão das métricas para melhor definição dos resultados, apresentando uma pequena melhoria nos desempenhos das predições ao realizar uma ação quando um drift é detectado.

Figura 12 – Resultados métricas modelo *Perceptron*




Fonte: elaborada pelo autor.

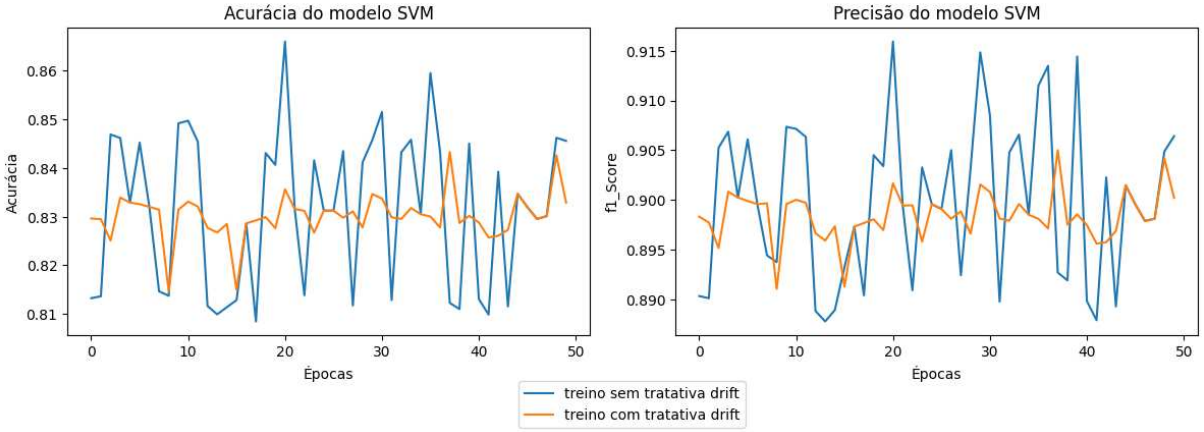
Tabela 3 – Análise de desempenho modelo *Perceptron*.

| Métrica | Média | Desvio Padrão |
|---|---|---|
| Acurácia (normal) | 0.838 | 0.1004 |
| Acurácia (*drift*) | 0.847 | 0.0160 |
| F1-Score (normal) | 0.895 | 0.1286 |
| F1-Score (*drift*) | 0.916 | 0.0116 |

Fonte: elaborada pelo autor.

Seguindo a mesma ideia definida no modelo anterior, alterando agora para o modelo SVM, nas mesmas 50 repetições, foram detectados 120 desvios. Obtivemos os seguintes resultados apresentados na Figura 13, onde a variação de cada métrica foi bem mais alta do que o modelo *Perceptron*. Podemos ver na Tabela 4 as médias e desvios dos desempenhos de cada métrica neste modelo, constantando uma diferença mínima, praticamente quase sem diferença entre as duas situações.

Figura 13 – Resultados métricas modelo *SVM*




Fonte: elaborada pelo autor.

Tabela 4 – Análise de desempenho modelo *SVM*.

| Métrica | Média | Desvio Padrão |
|---|---|---|
| Acurácia (normal) | 0.831 | 0.0156 |
| Acurácia (*drift*) | 0.830 | 0.0046 |
| F1-Score (normal) | 0.899 | 0.0078 |
| F1-Score (*drift*) | 0.898 | 0.0024 |

Fonte: elaborada pelo autor.

# 5 CONCLUSÕES E TRABALHOS FUTUROS

Esta pesquisa teve como objetivo realizar uma comparação das abordagens para o tratamento de *concept drifts* em dados processados com aprendizado de máquina. Ao longo do estudo, analisamos e avaliamos diferentes técnicas utilizadas para lidar com a ocorrência de mudanças de conceito nos dados de treinamento, o que pode comprometer a eficácia dos modelos de aprendizado.

Durante a revisão da literatura, foram identificadas algumas abordagens, tais como o retreinamento, retreinamento usando métodos ensemble e algoritmos adaptativos, que visam mitigar os efeitos negativos do *concept drift* nos modelos. Cada abordagem apresenta suas próprias vantagens e desvantagens, e a escolha da técnica mais adequada dependerá das características do problema em questão e dos requisitos específicos do projeto. Na pesquisa realizada foi utilizada a técnica de algoritmos adaptativos no modelo de treinamento do ML

Ao realizar a comparação das abordagens, foram considerados critérios como desempenho preditivo, capacidade de adaptação a mudanças de conceito e facilidade de implementação. Os resultados obtidos demonstraram que houve uma melhoria considerável na análise preditiva de um modelo, enquanto o outro a melhoria não foi tão significativa, provavelmente devido as características dos dados ou por conta de parâmetros de treinamento poucos trabalhados. Não existe uma solução única e universalmente superior para o tratamento de concept drifts, e a escolha da abordagem mais adequada dependerá do contexto específico.

Além disso, observamos que a detecção precoce e a monitorização contínua de concept drifts são aspectos fundamentais para garantir a robustez dos modelos de aprendizado de máquina. A utilização de técnicas de detecção de mudanças de conceito podem auxiliar na identificação oportuna dessas mudanças, permitindo a adoção de estratégias de tratamento adequadas.

Um dos desafios enfrentados na concepção dessa pesquisa foi a seleção do conjunto de dados que favorecesse a detecção dos desvios, já que o conjunto precisa apresentar comportamentos e características específicas que facilitassem a detecção do desvio, como por exemplo, possuir um caráter temporal de armazenamento dos dados e categorias relevantes para detecção. É importante ressaltar que a pesquisa nesse campo ainda está em constante evolução, e novas abordagens e técnicas estão sendo propostas regularmente. Portanto, é fundamental que os pesquisadores e profissionais da área estejam atualizados sobre os avanços mais recentes e continuem a investigar e aprimorar os métodos existentes.

Em suma, a comparação das abordagens para o tratamento de *concept drifts* em dados processados com ML é um tema relevante e desafiador. Embora não haja uma solução universalmente superior, esta pesquisa contribuiu para a compreensão das principais técnicas disponíveis e ressaltou a importância da detecção precoce e do monitoramento contínuo de mudanças de conceito. Com base nesses resultados, é possível orientar a seleção e implementação de abordagens mais adequadas em cenários reais, possibilitando o desenvolvimento de modelos de aprendizado mais robustos e eficazes.

# REFERÊNCIAS


BAYRAM, F.; AHMED, B. S.; KASSLER, A. From concept drift to model degradation: An overview on performance-aware drift detectors. **Knowledge-Based Systems**, Elsevier, v. 245, p. 108632, 2022.

BEGUM, A.; FATIMA, F.; SABAHATH, A. Implementation of deep learning algorithm with perceptron using tenzorflow library. In: IEEE. **2019 International Conference on Communication and Signal Processing (ICCSP)**. [*S. l.*], 2019. p. 0172–0175.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine learning**, Springer, v. 20, p. 273–297, 1995.

DATAAT. **Introdução ao Machine Learning**. 2022. Disponível em: https://dataat.github.io/introducao-ao-machine-learning/introdu%C3%A7%C3%A3o.html#machine-learning. Acesso em: 28 nov. 2022.

DATASCIENCEACADEMY. **Capítulo 6 – O Perceptron – Parte 1**. 2022. Disponível em: https://www.deeplearningbook.com.br/o-perceptron-parte-1/. Acesso em: 28 jun. 2023.

GEEKS. **Getting started with Machine Learning**. 2022. Disponível em: https://www.geeksforgeeks.org/getting-started-machine-learning/. Acesso em: 28 nov. 2022.

GEEKS. **ML | Stochastic Gradient Descent (SGD)**. 2023. Disponível em: https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/. Acesso em: 30 jun. 2023.

KAGGLE. **Repository of community published data and code**. 2022. Disponível em: https://www.kaggle.com/. Acesso em: 21 nov. 2022.

LU, J.; LIU, A.; DONG, F.; GU, F.; GAMA, J.; ZHANG, G. Learning under concept drift: A review. **IEEE Transactions on Knowledge and Data Engineering**, v. 31, n. 12, p. 2346–2363, 2019.

MAHESH, B. Machine learning algorithms-a review. **International Journal of Science and Research (IJSR).[Internet]**, v. 9, p. 381–386, 2020.

MOHAMMED, M.; KHAN, M. B.; BASHIER, E. B. M. **Machine Learning**. CRC Press, 2016. Disponível em: https://doi.org/10.1201/9781315371658.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. **IBM Journal of research and development**, IBM, v. 3, n. 3, p. 210–229, 1959.

SARKER, I. H. Machine learning: Algorithms, real-world applications and research directions. **SN Computer Science**, Springer Science and Business Media LLC, v. 2, n. 3, mar. 2021. Disponível em: https://doi.org/10.1007/s42979-021-00592-x.

VERGARA, S. C. **Projetos e relatorios de Pesquisa Em Administracao**. [*S. l.*]: Atlas, 2013.